Análisis de datos: ¿promedio o visualizar la distribución?

Guilherme Silveira
13/01/2021

En nuestra plataforma de cursos en línea, queremos mostrar a nuestros estudiantes actuales y potenciales lo satisfechos que están las personas después de estudiar un curso. Para eso, cada alumno da una nota entre 0 y 10, y podemos sacar el promedio. Por ejemplo, un curso con 4 alumnos que dieron notas [6, 8, 9, 10]tendría un promedio de 8.25.

Excelente, calculemos todos los promedios de nuestros cursos. Para esto usaré Pandas en Python y una conexión a una base de datos MySQL:

```
sql ='''select avg(nota) as media from Registros r
      where ...
      group by r.curso;'''
medias = pd.read_sql(sql, conexion)
```

Imprimimos el promedio, la moda y la mediana:

```
print(f'Média: {promedios.mean()}')   # Promedio: 9.03
print(f'Moda: {promedios.mode()}')    # Moda: 9
print(f'Mediana: {promedios.median()}') # Mediana: 9.06
```

Qué maravilloso, la nota promedio de nuestros cursos es aproximadamente 9, así como la nota que más aparece (moda, que parece no tener sentido en este análisis) es 9, y la mitad

de nuestros cursos (mediana) están sobre 9.06 y la mitad abajo. Pero, ¿cuánto me dice eso sobre mis cursos? Aún no mucho. La desviación estándar puede ayudarnos:

```
print(f'Desviacion estandar: {promedios.std()}')  #

Desviacion estandar: 0.33
```

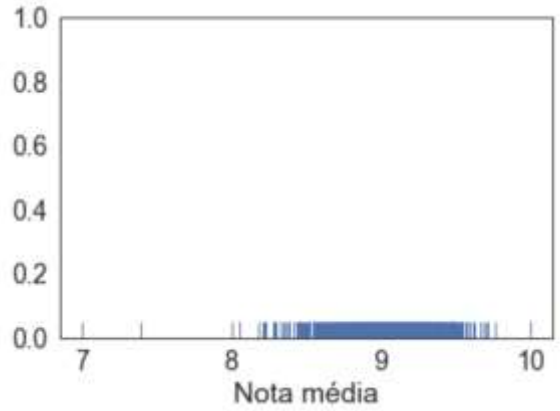
A y una desviación estándar indican que el 95% de nuestros cursos están entre 8,37 y 9,69. ¡Excelente! Éxito para la empresa.

Pero los números son difíciles de "visualizar"

En lugar de resumir todas las informaciones en un solo número, ¿vamos a intentar visualizar estos promedios?

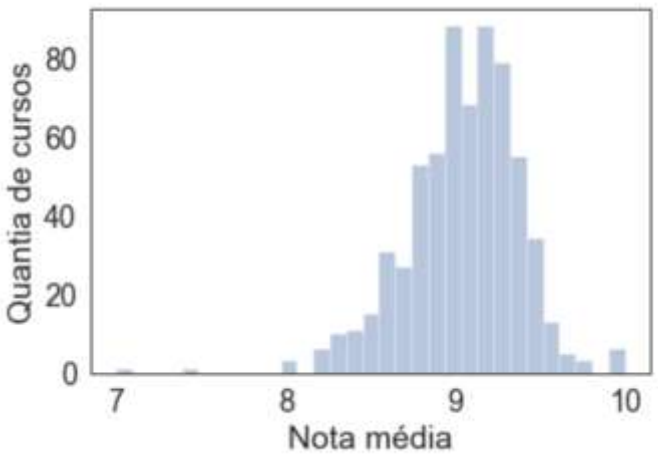
Usando la biblioteca seaborn (importada como `sns`) podemos trazar un" tick" para cada nota, un rugplot:

```
sns.rugplot(promedios)
```



Nota promedioPero como tenemos muchos "ticks" cercanos, no podemos ver la información correctamente. ¿Qué tal si acumulamos los ticks? Es decir, si dos cursos tienen nota entre 9 y 9,1, hago una barra de altura 2 ahí. Si cinco cursos tienen una nota promedio entre 9.1 y 9.2, hago una barra de altura 5 ahí. Estos son los conceptos básicos de un histograma: dividir el espacio de notas en diversos pequeños tramos y dibujar las barras de acuerdo:

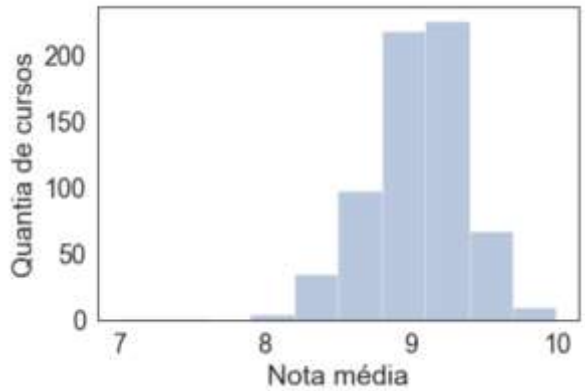
```
sns.distplot(promedios, hist=True, kde=False)
```



Cantidad de cursos x Nota promedio

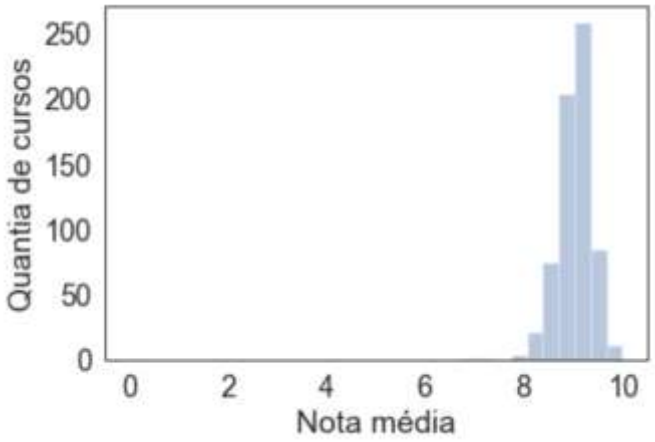
Ahora podemos ver que ningún curso tiene una nota promedio inferior a 7, y que el gran pastel de cursos está ahí con buenas notas entre 8.5 y 9.5. Pero podemos mejorar esta visualización, fíjate que la dividimos en muchas partes pequeñas, el propio seaborn intenta presuponer una división adecuada (hay algoritmos para eso), pero en algunos casos que nos parece interesante podemos forzar, ya que aquí quiero forzar 10 espacios (bins):

```
sns.distplot(promedios, hist=True, kde=False, bins=10)
```



Cantidad de cursos x Nota promedio

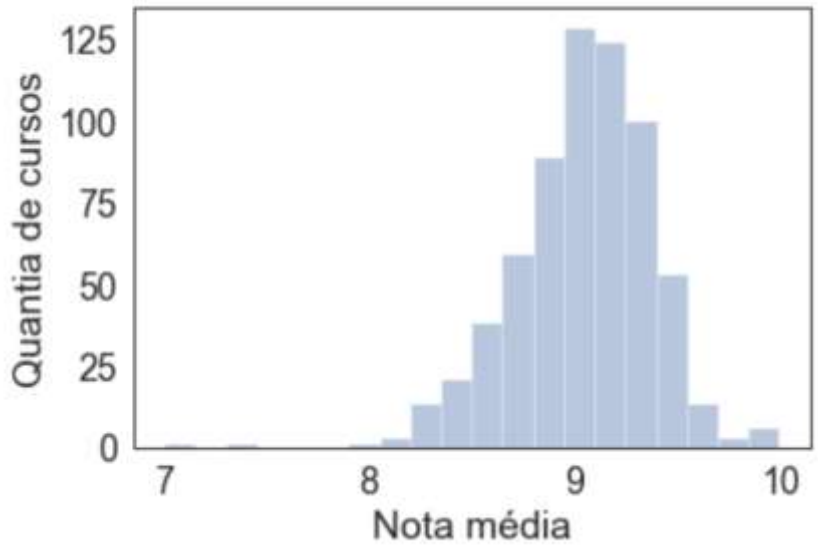
Pero tenga en cuenta que nuestras notas promedio para cada curso pueden variar entre 0 y 10, por lo que podemos establecer el valor mínimo y máximo, lo cual es genial para el ego, todas las notas promedio acumuladas allá a la derecha:



Cantidad de cursos x Nota promedio

Pero no ayuda entender la distribución de notas, ¿verdad? así que volvemos e intentalo solo con el eje x que sugiere la propia biblioteca, además de los 20 espacios ahora:

```
sns.distplot(promedios, hist=True, kde=False, bins=20)
```

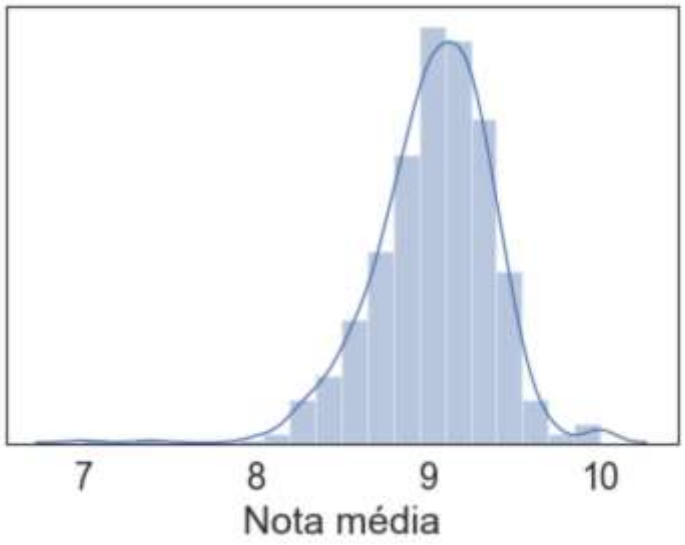


Cantidad de cursos x Nota promedio

Hay intentos de fórmula mágica para tratar de encontrar un excelente número de bins., y el estándar del seaborn intenta ayudarnos, pero ten en cuenta que siempre dependerá de la distribución de tus puntos y en lo que quieras centrarte. En nuestro caso, 20 parecía lo más interesante.

Por mucho que las barras del histograma den sentido a nuestros datos, sería interesante intentar trazar una función `f (promedio)` que detalle nuestros datos. La forma más tradicional de aproximar el histograma a una función es a través de un método llamado `kde`:

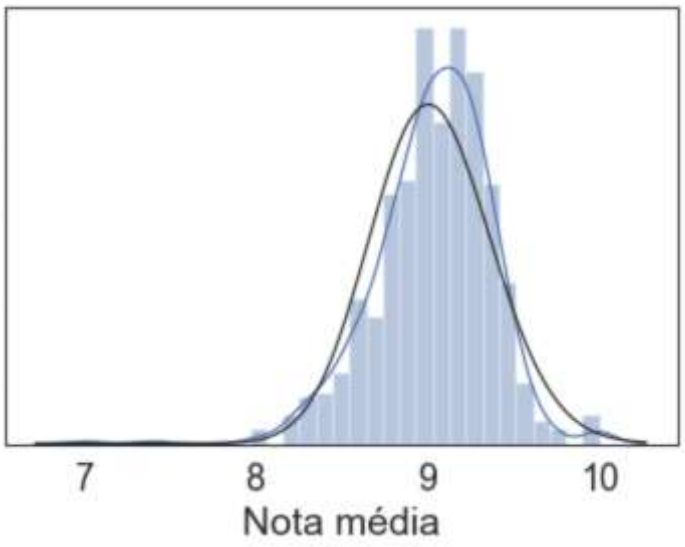
```
sns.distplot(promedios, hist=True, kde=True, bins=20)
```



Nota promedio

Esta función podría usarse para intentar predecir la distribución de puntos sobre los que no tenemos información. Observe cómo el comportamiento de distribución de esta función es muy similar a una distribución normal (gamma):

```
sns.distplot(promedios, fit=stats.gamma, hist=True, kde=True)
```



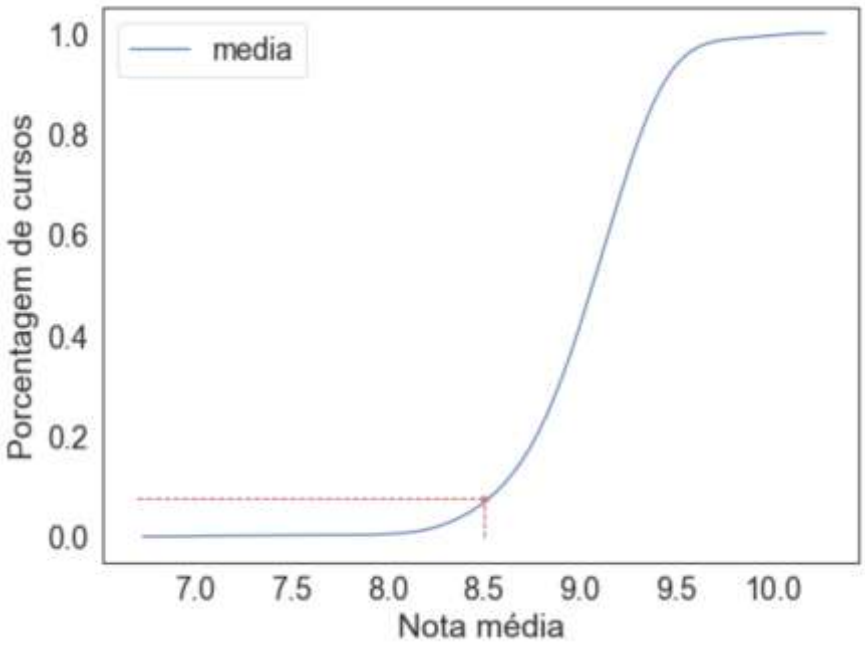
Nota promedio

Cuando sus datos se parecen a una normal de verdad (sin outliers, colas que tienden a cero, asimétricas, etc.), la medida del promedio del comienzo y la desviación estándar

pueden tener sentido, pero intentemos explorar un poco más lo que hemos conseguido. En general, más interesante que el histograma, es comprender el área debajo del histograma.

¿Qué podemos hacer? En lugar de plotar la función de distribución - en el punto 8, promedios iguales a 8, en el punto 9 promedios iguales a 9 - ¿vamos a acumular los valores? Es decir, en el punto 8 colocamos todas las notas menores que 8, en el punto 9, las menores que 9. Acumulando:

```
sns.kdeplot(promedios['promedio'], cumulative=True)
```



Porcentaje de cursos x Nota promedio

El gráfico acumulativo es muy importante. Muestra, por ejemplo, que si tomamos la nota ~ 8.5, tenemos que menos del 8% de los promedios están abajo. Además, solo ~ 10% de los promedios están sobre 9.5.

Aprovecho para detallar cómo hacer la anotación que hice en el gráfico, trazando una pequeña x y dos líneas:

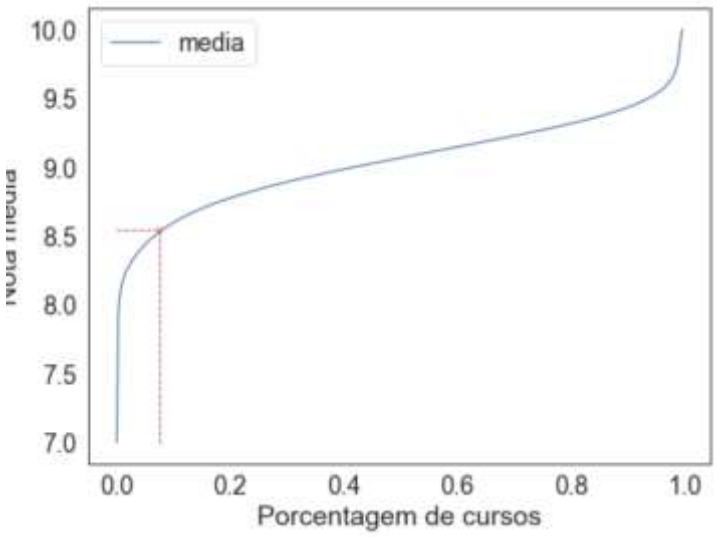
```
a4_dims = (8, 6)fig, ax = pyplot.subplots(figsize=a4_dims)
sns.kdeplot(promedios['promedio'], cumulative=True, ax=ax)

ax.set_xlabel('Nota promedio')
ax.set_ylabel('Porcentaje de cursos')
```

```
plt.plot(8.5, 0.075, "x")
plt.plot([8.5, 8.5], [0, 0.075], 'r--', lw=1)
plt.plot([6.7, 8.5], [0.075, 0.075], 'r--', lw=1)
```

Tres visualizaciones concluyentes

Pero, lo más interesante para nosotros será invertir la línea de pensamiento. Si rotamos nuestro gráfico, cambiando el eje x e y podemos ver los porcentajes acumulados y las notas:

```
sns.kdeplot(promedios['promedio'], cumulative=True, vertical=True)
```



Nota promedio x Porcentaje de cursos

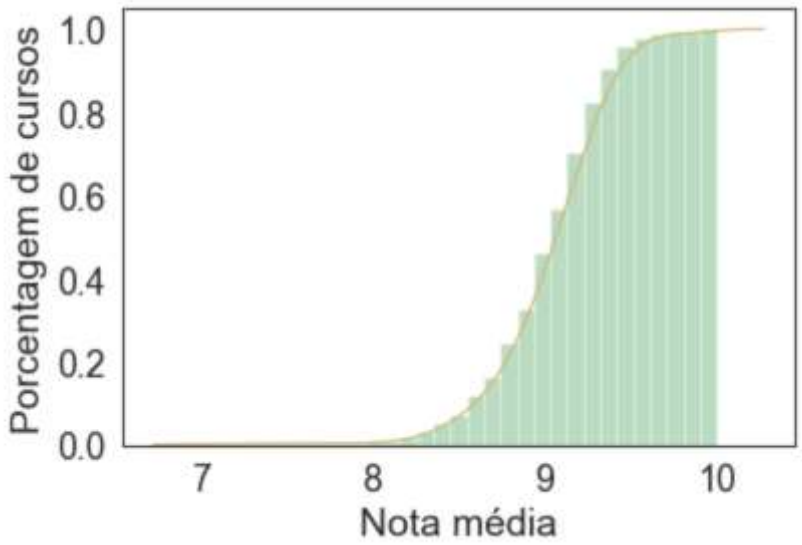
En esta vista rotada, es más fácil elegir un porcentaje, por ejemplo, 20% y darse cuenta de que solo el 20% de las notas están abajo de 8.8.

Reto para el lector: ¿cuál es la razón por la que, a largo plazo, es muy difícil (casi imposible) mantener un promedio máximo (10) en una evaluación?

Por fin, sin rotar nuestro gráfico, podemos trazar el histograma acumulativo junto a kde:

```
sns.distplot(promedios, hist=True, kde=True, color='y',
hist_kws={'cumulative': True, 'color':'g'},
```

```
kde_kws={'cumulative': True})
```



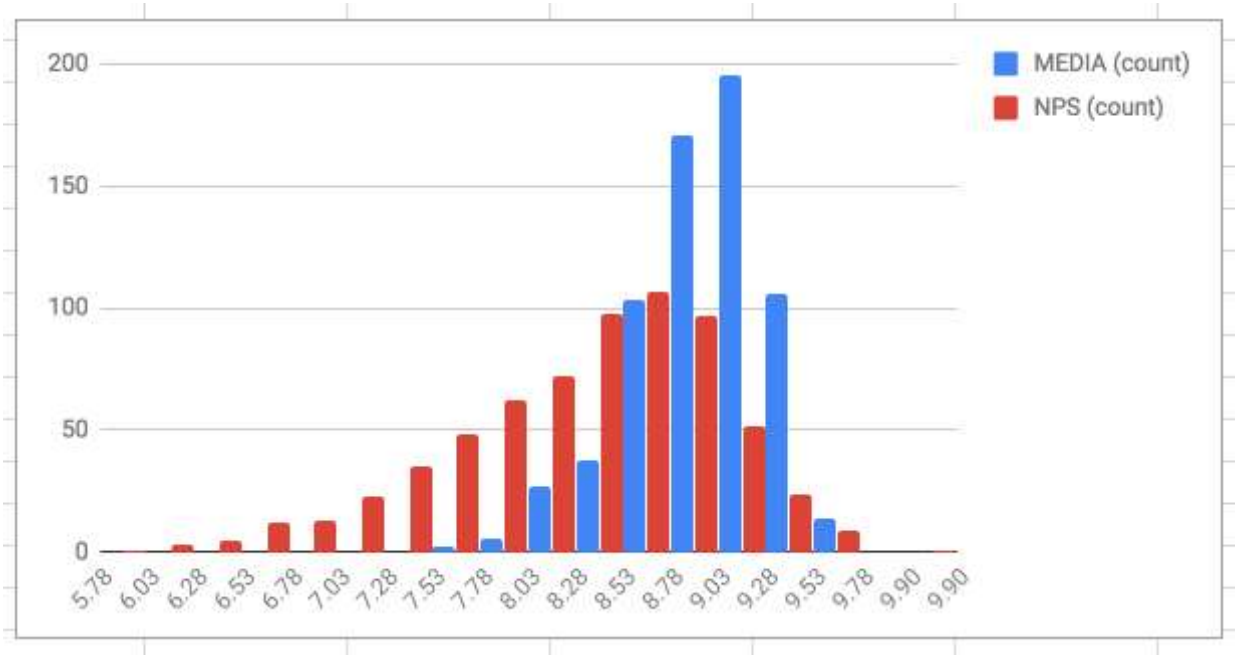
Porcentaje de cursos x Nota promedio

Nuestros dos últimos gráficos son excelentes herramientas para nosotros, como empresa, profesores y educadores, ya que indican que los alumnos en media terminan sus cursos de manera satisfactoria, con notas positivas.

Pero la media no parece ayudar un punto del alumno y de la alumna. Si los 10 mejores cursos tienen nota cercana a 10 y los buttom 10 tienen nota cercana a 8, la diferencia entre ellos no es tan clara. Y quienes estudian con nosotros les gustaría diferenciar claramente estos cursos, para saber qué esperar de un curso con una nota 8. Es por eso que la medida promedio que estamos usando para nuestros gráficos y que usábamos para mostrar la nota de nuestros cursos no parece ayudar.

Con el objetivo de ayudar el día a día de nuestros alumnos preferimos cambiar la nota visualizada a otra fórmula basada en el NPS (pero no exactamente NPS) para que la distribución de las notas quedara más "estirada" y nuestros alumnos y alumnas comprendan mejor que, aunque un curso tenga un alto número de alumnos y alumnas satisfechos, puede estar o no en nuestros top cursos. Con esta nueva fórmula la

distribución es:



Me gusta mucho el resumen de Paulo Silveira acerca de esta decisión:

“NPS genera números más bajos. Puede parecer raro, pero optamos por mostrar el 'NPS normalizado' como promedio porque creemos que estos números son más consistentes con la expectativa del alumno, con valores más bajos y diferenciados en cursos con notas más bajas. De lo contrario, el promedio simple pujaba el valor hacia arriba con facilidad, dada la abrumadora cantidad de alumnos satisfechos ".

Recursivamente dentro de la misma Alura tenemos un curso que enseña cómo trabajar con Python y Pandas, además de varios otros cursos para analizar nuestros datos y visualizaciones, dando el primer paso en su carrera como analista de datos.

Puedes leer también:

- ¿Media o mediana? Entienda cada una
- Análisis de datos: analizando mi distribución con tres alternativas de visualización
- Python: trabajando con diccionarios

ARTÍCULOS DE TECNOLOGÍA > DATA SCIENCE

BLOG

PROGRAMACIÓN
FRONT END
DATA SCIENCE
INNOVACIÓN Y GESTIÓN
DEVOPS

AOVS Sistemas de Informática S.A
CNPJ 05.555.382/0001-33

SÍGUENOS EN NUESTRAS REDES SOCIALES

ALIADOS

Empresa participante do SCALE UP ENDEAVOR

En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.

Google for Startups Alumni 2021

Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación

Lógica de Programación | Java

Cursos de Front End

HTML y CSS | JavaScript | React

Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps

Docker | Linux

Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento